Short and Deep: Sketching and Neural Networks

Amit Daniely, Nevena Lazic, Yoram Singer, Kunal Talwar[*]
Google Brain

Abstract

Data-independent methods for dimensionality reduction such as random projections, sketches, and feature hashing have become increasingly popular in recent years. These methods often seek to reduce dimensionality while preserving the hypothesis class, resulting in inherent lower bounds on the size of projected data. For example, preserving linear separability requires $\Omega(1/\gamma^2)$ dimensions, where γ is the margin, and in the case of polynomial functions, the number of required dimensions has an exponential dependence on the polynomial degree. Despite these limitations, we show that the dimensionality can be reduced further while maintaining performance guarantees, using improper learning with a slightly larger hypothesis class. In particular, we show that any sparse polynomial function of a sparse binary vector can be computed from a compact sketch by a single-layer neural network, where the sketch size has a *logarithmic* dependence on the polynomial degree. A practical consequence is that networks trained on sketched data are compact, and therefore suitable for settings with memory and power constraints. We empirically show that our approach leads to networks with fewer parameters than related methods such as feature hashing, at equal or better performance.

1 Introduction

In many supervised learning problems, input data are high-dimensional and sparse. The high dimensionality may be inherent in the domain, such as a large vocabulary in a language model, or the result of creating hybrid conjunction features. This setting poses known statistical and computational challenges for standard supervised learning techniques, as high-dimensional inputs lead to models with a very large number of parameters.

An increasingly popular approach to reducing model size is to map inputs to a lower-dimensional space in a data-independent manner, using methods such as random projections, sketches, and hashing. These mappings typically attempt to preserve the hypothesis class, leading to inherent theoretical limitations on size. For example, for linearly separable unit vectors with margin γ, it can be shown that at least $\Omega(1/\gamma^2)$ dimensions are needed to preserve linear separability, even if one can use arbitrary input embeddings (see Section D). It would therefore appear that data dimensionality cannot be reduced beyond this bound.

In this work, we show that using a slightly larger hypothesis class when decoding projections (improper learning) allows us to further reduce dimensionality while maintaining theoretical guarantees. In particular, we show that any sparse polynomial function of a sparse binary vector can be computed from a very compact sketch by a single-layer neural network. The hidden layer allows us to "decode" inputs from representations that are smaller than in existing work. In the simplest case, we show that for linearly separable k-sparse d-dimensional inputs, one can create a $O(k \log \frac{d}{\delta})$-dimensional sketch of the inputs and guarantee that a single-layer neural network can correctly classify $1-\delta$ fraction of the sketched data. In the case of polynomial functions, the required sketch size has a logarithmic dependence on the polynomial degree.

For binary k-sparse input vectors, we show that it suffices to have a simple feed-forward network with nonlinearity implemented via the commonly used rectified linear unit (Relu). We extend our results to real-valued data that is close to being k-sparse, using less conventional $\min$ and median nonlinearities. Furthermore, we show that data can be mapped using sparse sketching matrices.

[*]Email: kunal@google.com. Author for correspondences.

Thus, our sketches are efficient to compute and do not increase the number of non-zero input values by much, in contrast to standard dense Gaussian projections.

We empirically evaluate our sketches on real and synthetic datasets. Our approach leads to more compact neural networks than existing methods such as feature hashing and Gaussian random projections, at competitive or better performance. This makes our sketches appealing for deployment in settings with strict memory and power constraints, such as mobile and embedded devices.

2 Previous work

To put our work in context, we next summarize some lines of research related to this work.

Random projections and sketching. Random Gaussian projections are by now a standard tool for dimensionality reduction. For general vectors, the Johnson-Lindenstrauss (1984) Lemma implies that a random Gaussian projection into $O(\log(1/\delta)/\varepsilon^2)$ dimensions preserves the inner product between a pair of unit vectors up to an additive factor ε, with probability $1-\delta$. A long line of work has sought sparser projection matrices with similar guarantees; see (Achlioptas, 2003; Ailon & Chazelle, 2009; Matousek, 2008; Dasgupta et al., 2010; Braverman et al., 2010; Kane & Nelson, 2014; Clarkson & Woodruff, 2013). Research in streaming and sketching algorithms has addressed related questions. Alon et al. (1999) showed a simple hashing-based algorithm for unbiased estimators for the Euclidean norm in the streaming setting. Charikar et al. (2004) showed an algorithm for the heavy-hitters problem based on the *count sketch*. Most relevant to our works is the *count-min sketch* of Cormode and Muthukrishnan (2005a; 2005b).

Projections in learning. Random projections have been used in machine learning at least since the work of Arriaga and Vempala (2006). For fast estimation of a certain class of kernel functions, sampling has been proposed as a dimensionality reduction technique in (Kontorovich, 2007) and (Rahimi & Recht, 2007). Shi et al. (2009) propose using a count-min sketch to reduce dimensionality while approximately preserving inner products for sparse vectors. Weinberger et al. (2009) use the count-sketch to get an unbiased estimator for the inner product of sparse vectors and prove strong concentration bounds. Ganchev and Dredze (2008) empirically show that hashing is effective in reducing model size without significantly impacting performance. Hashing has also been used in Vowpal Wabbit (Langford et al., 2007). Talukdar and Cohen (2014) use the count-min sketch in graph-based semi-supervised learning. Pham and Pagh (2013) showed that a count sketch of a tensor power of a vector could be quickly computed without explicitly computing the tensor power, and applied it to fast sketching for polynomial kernels.

Compressive sensing. Our work is also related to compressive sensing. For k-sparse vectors, results in this area, e.g. (Donoho, 2006; Candés & Tao, 2006), imply that a k-sparse vector $\mathbf{x} \in \mathbb{R}^d$ can be reconstructed w.h.p. from a projection of dimension $O(k \ln \frac{d}{k})$. However, to our knowledge, no provable decoding algorithms are implementable by a low-depth neural network. Recent work by Mousavi et al. (2015) empirically explores using a deep network for decoding in compressive sensing and also considers learnt non-linear encodings to adapt to the distribution of inputs.

Parameter reduction in deep learning. Our work can be viewed as a method for reducing the number of parameters in neural networks. Neural networks have become ubiquitous in many machine learning applications, including speech recognition, computer vision, and language processing tasks(see (Hinton et al., 2012; Krizhevsky et al., 2012; Sermanet et al., 2013; Vinyals et al., 2014) for a few notable examples). These successes have in part been enabled by recent advances in scaling up deep networks, leading to models with millions of parameters (Dean et al., 2012; Krizhevsky et al., 2012). However, a drawback of such large models is that they are very slow to train, and difficult to deploy on mobile and embedded devices with memory and power constraints. Denil et al. (2013) demonstrate significant redundancies in the parameterization of several deep learning architectures, and they propose training low-rank decompositions of weight matrices. Cheng et al. (2015) impose circulant matrix structure on fully connected layers. Ba and Caruana (2014) train shallow networks to predict the log-outputs of a large deep network, and Hinton et al. (2015) train a small network to match smoothed predictions of a complex deep network or an ensemble of such models. Collins and Kohli (2014) encourage zero-weight connections using sparsity-inducing priors, while others such as LeCun et al. (1989); Hassibi et al. (1993); Han et al. (2015) use techniques for pruning weights. HashedNets (Chen et al., 2015) enforce parameter sharing between random groups of network parameters. In contrast to these methods, sketching only involves applying a sparse, linear projection to the inputs, and does not require a specialized learning procedure or network architecture.

3 Sketching

For simplicity, we first present our results for sparse binary vectors, and extend the discussion to real-valued vectors to Appendix A. Let $B_{d,k} = \{\mathbf{x} \in \{0,1\}^d : \|\mathbf{x}\|_0 \leq k\}$ be the set of k-sparse d-dimensional binary vectors. We sketch such vectors using a family of randomized sketching algorithms based on the count-min sketch, as described next.

Given a parameter m and a hash function $h : [d] \to [m]$, the sketch $S_h(\mathbf{x})$ of a vector $\mathbf{x} \in B_{d,k}$ is a binary vector $\mathbf{y}$ where each bit of $\mathbf{y}$ is the OR of bits of $\mathbf{x}$ that hash to it:

$$y_l = \bigvee_{i \,:\, h(i) = l} x_i \,.$$

We map data using a concatenation of several such sketches. Given an ordered set of hash functions $h_1, \ldots, h_t \overset{def}{=} h_{1:t}$, the sketch $S_{h_{1:t}}(\mathbf{x})$ is defined as a $m \times t$ matrix Y, where the j^{th} column corresponds to the sketch $S_{h_j}(\mathbf{x})$. We define the following procedure for decoding the i^{th} bit of the input $\mathbf{x}$ from its sketch Y:

$$D^{\text{AND}}_{h_{1:t}}(Y, i) \overset{def}{=} \bigwedge_{j \in [t]} Y_{h_j(i)j} \,. \tag{1}$$

Thus, the decoded bit i is simply the AND of the t bits of Y that index i hashes to in the sketch.

The following theorem summarizes an important property of these sketches. As a reminder, a set of hash functions $h_{1:t}$ from $[d]$ to $[m]$ is *pairwise independent* if for all $i \neq j \in [d]$ and $a, b \in [m]$, $\Pr[h(i) = a \wedge h(j) = b] = m^{-2}$.[1]

Theorem 3.1. *Let $\mathbf{x} \in B_{d,k}$ and for $j \in [t]$ let $h_j : [d] \to [m]$ be drawn uniformly and independently from a pairwise independent distribution with $m = ek$. Then for any i,*

$$\Pr[D^{\text{AND}}_{h_{1:t}}(S_{h_{1:t}}(\mathbf{x}), i) \neq x_i] \leq e^{-t} \,.$$

Proof. Fix a vector $\mathbf{x} \in B_{d,k}$. Let $\mathcal{E}_h(i) \overset{def}{=} \{i' \neq i : h(i') = h(i)\}$ denote the collision set of i for a particular hash function h. Decoding will fail if $x_i = 0$ and for each of the t hash functions, the collision set of i for contains an index of a non-zero bit of $\mathbf{x}$. For a particular h, the probability of this event is:

$$Pr\left[\bigvee_{i' \in \mathcal{E}_h(i)} x_{i'} = 1\right] \leq \sum_{i' : x_{i'} = 1} \Pr[h(i') = h(i)] \leq \frac{k}{m} = \frac{1}{e} \,,$$

where the second inequality follows since the sum is over at most k terms, and each term is m^{-1} by pairwise independence. Thus $\Pr[Y_{h_j(i)} \neq x_i] \leq e^{-1}$ for any $j \in [t]$. Since hash functions h_j are drawn independently, and decoding can fail only if all t hash functions fail, it follows that $\Pr[D^{\text{AND}}_{h_{1:t}}(S_{h_{1:t}}(\mathbf{x}), i) \neq x_i] \leq e^{-t}$. $\square$

Let $\mathcal{H}_{d,s}$ denote the set $\mathcal{H}_{d,s} = \{\mathbf{w} \in \mathbb{R}^d : \|\mathbf{w}\|_0 \leq s\}$. We have the following corollary:

Corollary 3.2. *Let $\mathbf{w} \in \mathcal{H}_{d,s}$ and $\mathbf{x} \in B_{d,k}$. For $t = \log(s/\delta)$, and $m = ek$, if $h_1, \ldots, h_t$ are drawn uniformly and independently from a pairwise independent distribution, then*

$$\Pr\left[\sum_i w_i \, D^{\text{AND}}_{h_{1:t}}(S_{h_{1:t}}(\mathbf{x}), i) \neq \mathbf{w}^\top \mathbf{x}\right] \leq \delta \,.$$

[1]Such hash families can be easily constructed (see e.g. Mitzenmacher & Upfal (2005)), using a $O(\log m)$-bit seed. Moreover, each hash can be evaluated using $O(1)$ arithmetic operations over $O(\log d)$-sized words.

4 Sparse linear functions

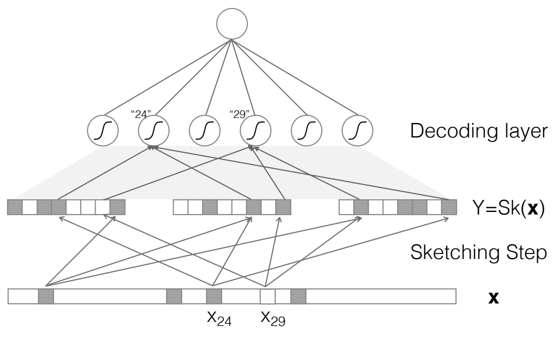

Figure 1: Neural-network sketching: sparse vector $\mathbf{x}$ maps to sketch using $t = 3$ hashes & $m = 8$; shaded squares designate 1's; sketching step is random; sketch then used as input to single-layer net: $\mathbf{w}^\top \mathbf{x}$; nodes labelled "24" & "29" correspond to decoding of x_{24} & x_{29} and shown with non-zero incoming edges.

Let $\mathbf{w} \in \mathcal{H}_{d,s}$, $\mathbf{x} \in B_{d,k}$, and $Y = S_{h_{1:t}}(\mathbf{x})$ for m, t satisfying the conditions of Corollary 3.2. We will now argue that there exists a one-layer neural network that takes Y as input and outputs $\mathbf{w}^\top \mathbf{x}$ with high probability (over the randomness of the sketching process).

Let $\mathcal{N}_n(f)$ denote the family of feed-forward neural networks with one hidden layer containing n nodes, nonlinearity f applied at each hidden unit, and a linear function at the output layer. We can construct a network in $\mathcal{N}_s(\mathrm{Relu})$ such that each hidden unit implements $D^{\mathrm{AND}}_{h_{1:t}}(Y, i)$ (i.e. decodes bit x_i from the sketch) for each index i in the support of $\mathbf{w}$. We can then set the output weights of the network to the corresponding non-zero weights w_i to get $\mathbf{w}^\top \mathbf{x}$.

It remains to show that a hidden unit can implement $D^{\mathrm{AND}}_{h_{1:t}}(Y, i)$. Indeed, the AND of t bits can be implemented using nearly any non-linearity. With $\mathrm{Relu}(a) = \max\{0, a\}$, we can construct the activation for bit x_i, $a_i = \sum_{(l,j)} V_{lj} Y_{lj} + B_i$, by setting the appropriate t weights in V_{lj} to 1, setting remaining weights to 0, and setting the bias B_i to $1 - t$. Using Corollary 3.2, we have the following theorem.

Theorem 4.1. *For every* $\mathbf{w} \in \mathcal{H}_{d,s}$ *there exists a set of weights for a network* $N \in \mathcal{N}_s(\mathrm{Relu})$ *such that for each* $\mathbf{x} \in B_{d,k}$*,*

$$Pr_{h_{1:t}}[N(S_{h_{1:t}}(\mathbf{x})) = \mathbf{w}^\top \mathbf{x}] \geq 1 - \delta\,,$$

as long as $m = ek$ *and* $t = \log(s/\delta)$*. Moreover, the weights coming into each node in the hidden layer are in* $\{0, 1\}$ *with at most t non-zeros.*

The final property implies that when using $\mathbf{w}$ as a linear classifier, we get small generalization error as long as the number of examples is at least $\Omega(s(1 + t \log mt))$. This can be proved, e.g., using standard compression arguments: each such model can be represented using only $st \log(mt)$ bits in addition to the representation size of $\mathbf{w}$. Similar bounds hold when we use ℓ_1 bounds on the weight coming into each unit. Note that even for $s = d$ (i.e. $\mathbf{w}$ is unrestricted), we get non-trivial input compression.

For comparison, we prove the following result for Gaussian projections in the appendix B. In this case, the model weights in our construction are not sparse.

Theorem 4.2. *For every* $\mathbf{w} \in \mathcal{H}_{d,s}$ *there exists a set of weights for a network* $N \in \mathcal{N}_s(\mathrm{Relu})$ *such that for each* $\mathbf{x} \in B_{d,k}$*,*

$$Pr_{h_{1:t}}[N(G\mathbf{x})) = \mathbf{w}^\top \mathbf{x}] \geq 1 - \delta\,,$$

as long as G is a random $m \times d$ *Gaussian matrix, with* $m \geq 4k \log(s/\delta)$*.*

5 Sparse polynomial functions

For boolean inputs, Theorem 4.1 extends immediately to sparse polynomial functions. Note that we can implement the AND of two bits $x_i \wedge x_j$ as the AND of the corresponding decodings $D^{\mathrm{AND}}_{h_{1:t}}(Y, i)$ and $D^{\mathrm{AND}}_{h_{1:t}}(Y, j)$. Since each decoding is an AND of t bits, the overall decoding is an AND of at most $2t$ locations in the sketch. More generally, we have the following theorem:

Theorem 5.1. *Given* $\mathbf{w} \in \mathbb{R}^s$*, and sets* $A_1, \ldots, A_s \subseteq [d]$*, let* $g : \{0,1\}^d \to \mathbb{R}$ *denote the polynomial*

$$g(\mathbf{x}) = \sum_{j=1}^{s} w_j \prod_{i \in A_j} x_i = \sum_{j=1}^{s} w_j \bigwedge_{i \in A_j} x_i\,.$$

Then there exists a set of weights for a network $N \in \mathcal{N}_s(\mathrm{Relu})$ such that for each $\mathbf{x} \in B_{d,k}$,

$$Pr_{h_{1:t}}[N(S_{h_{1:t}}(\mathbf{x})) = g(\mathbf{x})] \geq 1 - \delta\,,$$

as long as $m = ek$ and $t = \log(|\cup_{j \in [s]} A_j|/\delta)$. Moreover, the weights coming into each node in the hidden layer are in $\{0, 1\}$ with at most $t \cdot \left(\sum_{j\in[s]} |A_j|\right)$ non-zeros overall. In particular, when g is a degree-p polynomial, we can set $t = \log(ps/\delta)$, and each hidden unit has at most pt non-zero weights.

This is a setting where we get a significant advantage over proper learning. To our knowldege, there is no analog of this result for Gaussian projections. Classical sketching approaches would use a sketch of $\mathbf{x}^{\otimes p}$, which is a k^p-sparse vector over binary vectors of dimension d^p. Known sketching techniques such as Pham & Pagh (2013) would construct a sketch of size $\Omega(k^p)$. Practical techniques such as Vowpal Wabbit also construct cross features by explicitly building them and have this exponential dependence. In stark contrast, neural networks allow us to get away with a logarithmic dependence on p.

Using polynomial kernels. Theorems 4.1 has a corresponding variants where the neural net is replaced by a polynomial of degree t. Similarly, the neural net in Theorem 5.1 can be replaced by a degree-pt polynomial when the polynomial g has degree p. This implies that one can use a polynomial kernel to get efficient learning.

Deterministic sketching. A natural question that arises is whether the parameters above can improved. We show in App. C that if we allow large scalars in the sketches, one can construct a deterministic $(2k + 1)$-dimensional sketch from which a shallow network can reconstruct any monomial. We also show a lower bound of k on the required dimensionality.

Lower bound for proper learning. We can also show, see App. D, that if one does not expand the hypothesis class, then even in the simplest of settings of linear classifiers over 1-sparse vectors, the required dimensionality of the projection is much larger than the dimension needed for improper learning. The result is likely folklore and thus we present a short proof in the appendix for completeness using concrete constants in the theorem and its proof below.

Neural nets on Boolean inputs. We remark that for Boolean inputs (irrespective of sparsity), any polynomial with s monomials can be represented by a neural network in $\mathcal{N}_s(\mathrm{Relu})$ using the construction in Theorem 5.1.

6 Experiments with synthetic data

In this section, we evaluate sketches on synthetically generated datasets for the task of polynomial regression. In all the experiments here, we assume input dimension $d = 10^4$, input sparsity $k = 50$, hypothesis support $s = 300$, and $n = 2 \times 10^5$ examples. We assume that only a subset of features $\mathcal{I} \subseteq [d]$ are relevant for the regression task, with $|\mathcal{I}| = 50$. To generate an hypothesis, we select s subsets of relevant features $A_1, \ldots, A_s \subset \mathcal{I}$ each of cardinality at most 3, and generate the corresponding weight vector $\mathbf{w}$ by drawing corresponding s non-zero entries from the standard Gaussian distribution. We generate binary feature vectors $\mathbf{x} \in B_{d,k}$ as a mixture of relevant and other features. Concretely, for each example we draw 12 feature indices uniformly at random from $\mathcal{I}$, and the remaining indices from $[d]$. We generate target outputs as $g(\mathbf{x}) + z$, where $g(\mathbf{x})$ is in the form of the polynomial given in Theorem 5.1, and z is additive Gaussian noise with standard deviation 0.05. In all experiments, we train on 90% of the examples and evaluate mean squared error on the rest.

We first examined the effect of the sketching parameters m (hash size) and t (number of hash functions) on sparse linear regression error. We generated synthetic datasets as described above (with all feature subsets in $\mathcal{A}$ having cardinality 1) and trained networks in $\mathcal{N}_s(\mathrm{Relu})$. The results are shown in Figure 2 (left). As expected, increasing t leads to better performance. Using hash size m less than the input sparsity k leads to poor results, while increasing hash size beyond ek (in this case, $ek \approxeq 136$) for reasonable t yields only modest improvements.

We next examined the advantages of improper learning. We generated 10 sparse linear regression datasets and trained linear models and networks in $\mathcal{N}_s(\mathrm{Relu})$ on original and sketched features with

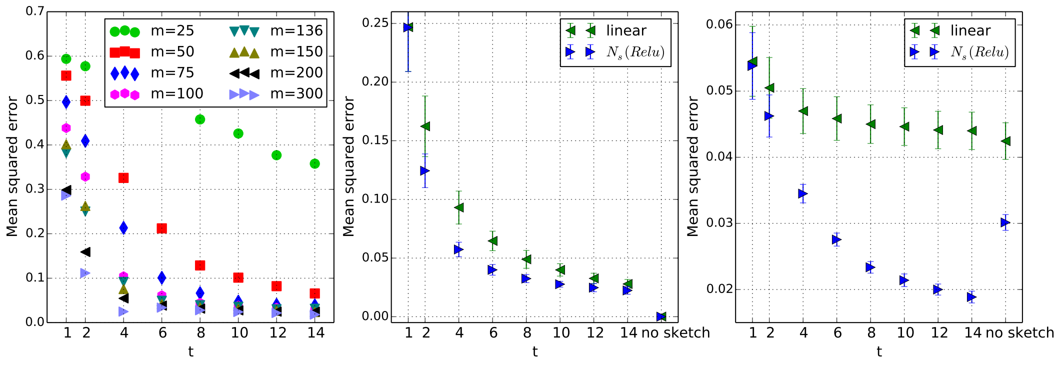

Figure 2: Left: effect of varying t, m for sketched 1-hidden layer network. Center: sparse linear regression on sketched data with improper learning. Right: sparse polynomial regression on sketched data.

$m = 200$ and several values of t. The results are shown in Figure 2 (center). The neural network yields notably better performance than a linear model. This suggests that linear classifiers are not well-preserved after projections, as the $\Omega(1/\gamma^2)$ projection size required for linear separability can be large. Applying a neural network to sketched data allows us to use smaller projections.

Table 1: Comparison of sketches and Gaussian random projections on the sparse linear regression task (top) and sparse polynomial regression task (bottom). See text for details.

	1K	2K	3K
Gaussian	0.089	0.057	0.029
Sketch $t = 1$	0.087	0.049	0.031
Sketch $t = 2$	0.072	0.041	0.023
Sketch $t = 6$	0.041	0.033	0.022
Gaussian	0.043	0.037	0.034
Sketch $t = 1$	0.041	0.036	0.033
Sketch $t = 2$	0.036	0.027	0.024
Sketch $t = 6$	0.032	0.022	0.018

We repeated the previous experiment for 10 polynomial regression datasets, generated with feature subsets in $\mathcal{A}$ of cardinality 2 and 3. The results are shown in Figure 2 (right). The linear model is a bad fit, showing that $g(\mathbf{x})$ is not well approximated by a linear function. Neural networks applied to sketches succeed in learning a small model and achieve significantly lower error than a network applied to the original features for $t \geq 6$. This suggests that reducing the input size, and consequently the number of model parameters, can lead to better generalization. Note that previous work on hashing and projections would imply using significantly larger sketch size for this setting.

We also compared our sketches to Gaussian random projections. We generated sparse linear and polynomial regression datasets with the same settings as before, and reduce the dimensionality of the inputs to 1000, 2000 and 3000 using Gaussian random projections and sketches with $t \in \{1, 2, 6\}$. We remark that in this comparison, the column headings correspond to the total sketch size mt. Thus, e.g., when we take $t = 6$, m is correspondingly reduced. We report the squared error averaged across examples and five datasets of one-layer neural networks in Table 1. The results demonstrate that sketches with $t > 1$ yield lower error than Gaussian projections. Note also that Gaussian projections are dense and hence much slower to train.

7 Experiments with Language Processing Tasks

Linear and low degree sparse polynomials are often used for classification. Our results imply that if we have linear or a sparse polynomial with classification accuracy $1-\varepsilon$ over some set of examples in $B_{d,k} \times \{0, 1\}$, then neural networks constructed to compute the linear or polynomial function attain accuracy of at least $1-\varepsilon-\delta$ over the same examples. Moreover, the number of parameters in the new network is relatively small by enforcing sparsity or ℓ_1 bounds for the weights into the hidden layers. We thus get generalization bounds with negligible degradation with respect to non-sketched predictor. In this section, we evaluate sketches on the language processing classification tasks described below.

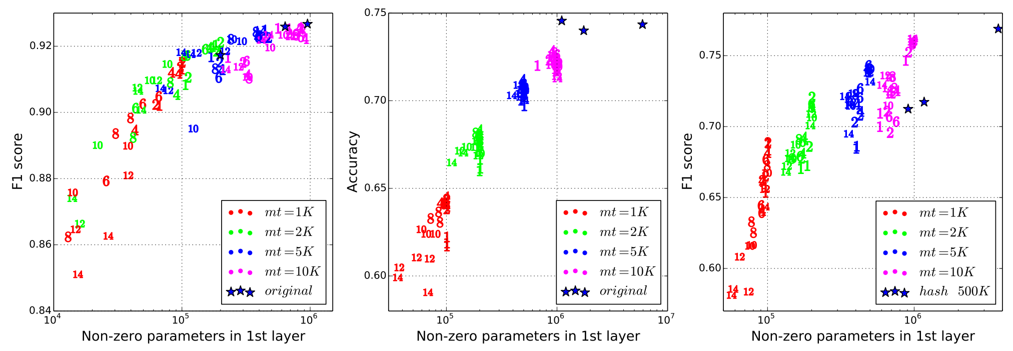

Figure 3: Performance vs. number of nonzero parameters in 1st layer for Reuters (left), AG News (center), and type tagging (right). Each color corresponds to a different sketch size and markers indicate the number of subsketches t. We evaluate each setting for three values of the l_1 regularization parameter λ_1.

Entity Type Tagging. Entity type tagging is the task of assigning one or more labels (such as *person*, *location*, *organization*, *event*) to mentions of entities in text. We perform type tagging on a corpus of new documents containing 110K mentions annotated with 88 labels (on average, 1.7 labels per mention). Features for each mention include surrounding words, syntactic and lexical patterns, leading to a very large dictionary. Similarly to previous work, we map each string feature to a 32 bit integer, and then further reduce dimensionality using hashing or sketches. See Gillick et al. (2014) for more details on features and labels for this task.

Reuters-news Topic Classification. The Reuters RCV1 data set consists of a collection of approximately 800,000 text articles, each of which is assigned multiple labels. There are 4 high-level categories: Economics, Commerce, Medical, and Government (ECAT, CCAT, MCAT, GCAT), and multiple more specific categories. We focus on training binary classifiers for each of the four major categories. The input features we use are binary unigram features. Post word-stemming, we get data of approximately 113,000 dimensions. The feature vectors are very sparse, however, and most examples have fewer than 120 non-zero features.

AG-news Topic Classification. We perform topic classification on $680K$ articles from AG news corpus, labeled with one of 8 news categories: *Business*, *Entertainment*, *Health*, *Sci/Tech*, *Sports*, *Europe*, *U.S.*, *World*. For each document, we extract binary word indicator features from the title and description; in total, there are 210K unique features, and on average, 23 non-zero features per document.

Experimental Setup. In all experiments, we use two-layer feed-forward networks with ReLU activations and 100 hidden units in each layer. We use a softmax output for multiclass classification and multiple binary logistic outputs for multilabel tasks. We experimented with input sizes of 1000, 2000, 5000, and 10,000 and reduced the dimensionality of the original features using sketches with $t \in \{1, 2, 4, 6, 8, 10, 12, 14\}$ blocks. In addition, we experimented with networks trained on the original features. We encouraged parameter sparsity in the first layer using ℓ_1-norm regularization and learn parameters using the proximal stochastic gradient method. As before, we trained on 90% of the examples and evaluated on the remaining 10%. We report accuracy values for multiclass classification, and F1 score for multilabel tasks, with true positive, false positive, and false negative counts accumulated across all labels.

Results. Since one motivation for our work is reducing the number of parameters in neural network models, we plot the performance metrics versus the number of non-zero parameters in the first layer of the network. The results are shown in Figure 3 for different sketching configurations and settings of the ℓ_1-norm regularization parameters (λ_1). On the entity type tagging task, we compared sketches to a single hash function of size 500,000 as the number of the original features is too large. In this case, sketching allows us to both improve performance and reduce the number of parameters. On the Reuters task, sketches achieve similar performance to the original features with fewer parameters. On AG news, sketching results in more compact models at a modest drop in accuracy. In almost all cases, multiple hash functions yield higher accuracy than a single hash function for similar model size.

8 Conclusions

We have presented a simple sketching algorithm for sparse boolean inputs, which succeeds in significantly reducing the dimensionality of inputs. A single-layer neural network on the sketch can provably model any sparse linear or polynomial function of the original input. For k-sparse vectors in $\{0,1\}^d$, our sketch of size $O(k \log s/\delta)$ allows computing any s-sparse linear or polynomial function on a $1-\delta$ fraction of the inputs. The hidden constants are small, and our sketch is sparsity preserving. Previous work required sketches of size at least $\Omega(s)$ in the linear case and size at least k^p for preserving degree-p polynomials. Our results can be viewed as showing a compressed sensing scheme for 0-1 vectors, where the decoding algorithm is a depth-1 neural network. Our scheme requires $O(k \log d)$ measurements, and we leave open the question of whether this can be improved to $O(k \log \frac{d}{k})$ in a stable way. We demonstrated empirically that our sketches work well for both linear and polynomial regression, and that using a neural network does improve over a direct linear regression. We show that on real datasets, our methods lead to smaller models with similar or better accuracy for multiclass and multilabel classification problems. In addition, the compact sketches lead to fewer trainable parameters and faster training.

Acknowledgements

We would like to thank Amir Globerson for numerous fruitful discussion and help with an early version of the manuscript.

References

Dimitris Achlioptas. Database-friendly random projections: Johnson–Lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.

N. Ailon and B. Chazelle. The fast JL transform and approximate nearest neighbors. *SICOMP*, 39(1), 2009.

Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

Rosa I. Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning*, 63(2):161–182, 2006.

J. Ba and R. Caruana. Do deep nets really need to be deep? In *NIPS*, pp. 2654–2662, 2014.

K. Do Ba, P. Indyk, E. Price, and D. Woodruff. Lower bounds for sparse recovery. In *SODA*, 2010.

A. Barron. Approximation and estimation bounds for artificial neural networks. *Mach. Learning*, 14(1), 1994.

Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *Information Theory, IEEE Transactions on*, 39(3):930–945, 1993.

Vladimir Braverman, Rafail Ostrovsky, and Yuval Rabani. Rademacher chaos, random Eulerian graphs and the sparse Johnson-Lindenstrauss transform. *CoRR*, abs/1011.2590, 2010.

E.J. Candés and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

M. Charikar, K. Chen, and M. Farach. Finding frequent items in data streams. *Theor. Comp. Sci.*, 312(1), 2004.

Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing convolutional neural networks. *CoRR*, abs/1506.04449, 2015. URL `http://arxiv.org/abs/1506.04449`.

Y. Cheng, F. Yu, r. Feris, S. Kumar, A. Choudhary, and S-F. Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *CVPR*, pp. 2857–2865, 2015.

K. Clarkson and D. Woodruff. Low rank approximation and regression in input sparsity time. In *STOC*, 2013.

M.D. Collins and P. Kohli. Memory bounded deep convolutional networks. *CoRR*, abs/1412.1442, 2014.

G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005a.

G. Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. In *SDM*, pp. 44–55, 2005b.

Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse Johnson–Lindenstrauss transform. In *STOC*, pp. 341–350. ACM, 2010.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V Le, MarcAurelio Ranzato, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pp. 1223–1231, 2012.

Misha Denil, Babak Shakibi, Laurent Dinh, MarcAurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pp. 2148–2156, 2013.

David L Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

Kuzman Ganchev and Mark Dredze. Small statistical models by random feature mixing. In *Workshop on Mobile NLP at ACL*, 2008.

D. Gillick, N. Lazic, K. Ganchev, J. Kirchner, and D. Huynh. Context-dependent fine-grained entity type tagging. *CoRR*, abs/1412.1820, 2014. URL `http://arxiv.org/abs/1412.1820`.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pp. 1135–1143, 2015.

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 89, 1993.

G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *CoRR*, 1503.02531, 2015.

W.B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math*, 26, 1984.

Daniel M. Kane and Jelani Nelson. Sparser Johnson–Lindenstrauss transforms. *J. ACM*, 61(1):4:1–4:23, 2014.

Leonid Kontorovich. A universal kernel for learning regular languages. In *MLG*, 2007.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

J. Langford, L. Li, and A. Strehl. Vowpal Wabbit online learning project. http://hunch.net/$\sim$vw/, 2007.

Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 89, 1989.

J. Matousek. On variants of the Johnson–Lindenstrauss lemma. *Rand. Struct. Algs.*, 33(2):142–156, 2008.

Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005. ISBN 0521835402.

Ali Mousavi, Ankit B. Patel, and Richard G. Baraniuk. A deep learning approach to structured signal recovery. arXiv:1508.04065, 2015.

Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *KDD*, pp. 239–247. ACM, 2013.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, pp. 1177–1184, 2007.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv:1312.6229*, 2013.

Q. Shi, J. Petterson, G. Dror, J. Langford, A. J. Smola, A. Strehl, and V. Vishwanathan. Hash kernels. In *Artificial Intelligence and Statistics AISTATS'09*, Florida, April 2009.

P.P. Talukdar and W.W. Cohen. Scaling graph-based semi supervised learning to large number of labels using count-min sketch. In *AISTATS*, volume 33 of *JMLR Proceedings*, pp. 940–947. JMLR.org, 2014.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014. URL `http://arxiv.org/abs/1411.4555`.

K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, and A. J. Smola. Feature hashing for large scale multitask learning. In *International Conference on Machine Learning*, 2009.

A General sketches

The results of Section 3 and Section 4 extend naturally to positive and to general real-valued vectors. We denote by

$$head_k(\mathbf{x}) = \underset{\mathbf{y}\,:\,\|\mathbf{y}\|_0 \le k}{\arg\min} \|\mathbf{x} - \mathbf{y}\|_1$$

the closest vector to $\mathbf{x}$ whose support size is k and the residue $tail_k(\mathbf{x}) = \mathbf{x} - head_k(\mathbf{x})$. Let $\mathbb{R}^+_{d,k,c}$ represent the set

$$\mathbb{R}^+_{d,k,c} = \{\mathbf{x} \in \mathbb{R}^d_+ : \|tail_k(\mathbf{x})\|_1 \le c\},$$

and let $\mathbb{R}_{d,k,c}$ represent the set

$$\mathbb{R}_{d,k,c} = \{\mathbf{x} \in \mathbb{R}^d : \|tail_k(\mathbf{x})\|_1 \le c\}.$$

For vectors in $\mathbb{R}^+_{d,k,c}$, we use the following sketching and decoding procedures analogous to the binary case.

$$S_{h_{1:t}}(\mathbf{x}) = [\mathbf{y}_1, ..., \mathbf{y}_t] \quad \text{where } y_l = \sum_{i:h(i)=l} x_i$$

$$D^{\text{MIN}}_{h_{1:t}} \overset{def}{=} \min_{j\in[t]}(y_{h_j(i)}).$$

Theorem A.1. *Let $\mathbf{x} \in \mathbb{R}^+_{d,k,c}$ and let $h_1, \ldots, h_t$ be drawn uniformly and independently from a pairwise independent distribution, for $m = e(k + \frac{1}{\varepsilon})$. Then for any i,*

$$\Pr\left[D^{\text{MIN}}_{h_{1:t}}(S_{h_{1:t}}(\mathbf{x}), i) \notin [x_i, x_i + \varepsilon c]\right] \le e^{-t}.$$

Proof. Fix a vector $\mathbf{x} \in \mathbb{R}^+_{d,k,c}$. To remind the reader, for a specific i and h, we have defined the set of collision indices $\mathcal{E}(i) \overset{def}{=} \{i' \ne i : h(i') = h(i)\}$. Note that $D^{\text{MIN}}_h(S_h(\mathbf{x}), i) = y_{h(i)}$, and we can rewrite $y_{h(i)}$ as $y_{h(i)} = x_i + \sum_{i'\in\mathcal{E}(i)} x_{i'} \ge x_i$. We next show that

$$\Pr\left[D^{\text{MIN}}_h(S_h(\mathbf{x}), i) > x_i + \varepsilon c\right] \le 1/e. \tag{2}$$

We can rewrite

$$D^{\text{MIN}}_h(S_h(\mathbf{x}), i) - x_i = \sum_{i'\in\mathcal{S}(head_k(\mathbf{x}))\cap\mathcal{E}(i)} x_{i'} + \sum_{i'\in\mathcal{S}(tail_k(\mathbf{x}))\cap\mathcal{E}(i)} x_{i'}.$$

By definition of $tail_k(\cdot)$, the expectation of the second term is c/m. Using Markov's inequality, we get that

$$\Pr[\sum_{i'\in\mathcal{S}(tail_k(\mathbf{x}))\cap\mathcal{E}(i)} x_{i'} > \varepsilon c] \le \frac{1}{\varepsilon m}.$$

To bound the first term, note that

$$\Pr\left[\sum_{i'\in\mathcal{S}(head_k(\mathbf{x}))\cap\mathcal{E}(i)} x_{i'} \ne 0\right] \le \Pr\left[\sum_{i'\in\mathcal{S}(head_k(\mathbf{x}))\cap\mathcal{E}(i)} 1 \ne 0\right] \le \mathbf{E}\left[\sum_{i'\in\mathcal{S}(head_k(\mathbf{x}))\cap\mathcal{E}(i)} 1\right] \le \frac{k}{m}.$$

Recall that $m = e(k + 1/\varepsilon)$, then, using the union bound establishes (2). The rest of the proof is identical to Theorem 3.1. ☐

Corollary A.2. *Let $\mathbf{w} \in \mathcal{H}_{d,s}$ and $\mathbf{x} \in \mathbb{R}^+_{d,k,c}$. For $t = \log(s/\delta)$, and $m = e(k + \frac{1}{\varepsilon})$, if $h_1, \ldots, h_t$ are drawn uniformly and independently from a pairwise independent distribution, then*

$$\Pr\left[|\sum_i w_i \cdot D^{\text{MIN}}_{h_{1:t}}(S_{h_{1:t}}(\mathbf{x}), i) - \mathbf{w}^\top\mathbf{x}| \ge \varepsilon c\|\mathbf{w}\|_1\right] \le \delta.$$

The proof in the case that $\mathbf{x}$ may not be non-negative is only slightly more complicated. Let us define the following additional decoding procedure,

$$D^{\text{MED}}_{h_{1:t}}(Y, i) \overset{def}{=} \operatorname*{Median}_{j\in[t]} Y_{h_j(i),j}.$$

Theorem A.3. *Let $\mathbf{x} \in \mathbb{R}_{d,k,c}$, and let $h_1, \ldots, h_t$ be drawn uniformly and independently from a pairwise independent distribution, for $m = 4e^2(k + 2/\varepsilon)$. Then for any i,*

$$\Pr\left[D^{\text{MED}}_{h_{1:t}}(S_{h_{1:t}}(\mathbf{x}), i) \notin [x_i - \varepsilon c, x_i + \varepsilon c]\right] \le e^{-t}.$$

Proof. As before, fix a vector $\mathbf{x} \in \mathbb{R}_{d,k,c}$ and a specific i and h. We once again write

$$D_h^{\mathrm{MED}}(S_h(\mathbf{x}), i) - x_i \;=\; \sum_{i' \in \mathcal{S}(head_k(\mathbf{x})) \cap \mathcal{E}(i)} x_{i'} \;+\; \sum_{i' \in \mathcal{S}(tail_k(\mathbf{x})) \cap \mathcal{E}(i)} x_{i'} \;.$$

By the same argument as the proof of Theorem A.1, the first term is nonzero with probability k/m. The second term has expectation in $[-c/s, c/s]$. Once again by Markov's inequality,

$$\Pr\left[\sum_{i' \in \mathcal{S}(tail_k(\mathbf{x})) \cap \mathcal{E}(i)} x_{i'} > \varepsilon c\right] \leq \frac{1}{\varepsilon m} \quad \text{and} \quad \Pr\left[\sum_{i' \in \mathcal{S}(tail_k(\mathbf{x})) \cap \mathcal{E}(i)} x_{i'} < -\varepsilon c\right] \leq \frac{1}{\varepsilon m}\,.$$

Recalling that $m = 4e^2(k + 2/\varepsilon)$, a union bound establishes that for any j,

$$\Pr[|D_{h_j}^{\mathrm{MED}}(S_{h_j}(\mathbf{x}), i) - x_i| > \varepsilon c] \leq \frac{1}{4e^2}.$$

Let X_j be indicator for the event that $|D_{h_j}^{\mathrm{MED}}(S_{h_j}(\mathbf{x}), i) - x_i| > \varepsilon c$. Thus $X_1, \ldots, X_t$ are binomial random variables with $Pr[X_j = 1] \leq \frac{1}{4e^2}$. Then by Chernoff's bounds

$$\begin{aligned}
&\Pr\left[D_{h_{1:t}}^{\mathrm{MED}}(S_{h_{1:t}}(\mathbf{x}), i) \notin [x_i - \varepsilon c, x_i + \varepsilon c]\right] \\
&\leq Pr\left[\sum_j X_j > t/2\right] \\
&\leq \exp\left(-\left(\frac{1}{2}\ln\frac{1/2}{1/4e^2} + \frac{1}{2}\ln\frac{1/2}{1 - 1/4e^2}\right)t\right) \\
&\leq \exp\left(-\left(\frac{1}{2}\ln 2e^2 + \frac{1}{2}\ln\frac{1}{2}\right)t\right) \\
&\leq \exp(-t)\,.
\end{aligned}$$

□

Corollary A.4. *Let $\mathbf{w} \in \mathcal{H}_{d,s}$ and $\mathbf{x} \in \mathbb{R}_{d,k,c}$,. For $t = \log(s/\delta)$, and $m = 4e^2(k + \frac{1}{\varepsilon})$, if $h_1, \ldots, h_t$ are drawn uniformly and independently from a pairwise independent distribution, then*

$$\Pr\left[\left|\sum_i w_i\, D_{h_{1:t}}^{\mathrm{MED}}(S_{h_{1:t}}(\mathbf{x}), i) - \mathbf{w}^\top \mathbf{x}\right| \geq \varepsilon c \|\mathbf{w}\|_1\right] \leq \delta\,.$$

To implement $D^{\mathrm{MIN}}(\cdot)$ using hidden units in a neural network, we need to use a slightly non-conventional non-linearity. For a weight vector $\mathbf{z}$ and input vector $\mathbf{x}$, a $\min$ gate implements $\min_{i: z_i \neq 0} z_i x_i$. Then, using Corollary A.2, we get the following theorem.

Theorem A.5. *For every $\mathbf{w} \in \mathcal{H}_{d,s}$ there exists a set of weights for a network $N \in \mathcal{N}_s(\min)$ such that for each $\mathbf{x} \in \mathbb{R}^+_{d,k,c}$ the following holds,*

$$Pr_{h_{1:t}}\left[\,|N(S_{h_{1:t}}(\mathbf{x})) - \mathbf{w}^\top \mathbf{x}| \geq \varepsilon c \|\mathbf{w}\|_1\,\right] \leq \delta\,.$$

as long as $m = e(k + \frac{1}{\varepsilon})$ and $t = \log(s/\delta)$. Moreover, the weights coming into each node in the hidden layer are binary with t non-zeros.

For real vectors $\mathbf{x}$, the non-linearity needs to implement a median. Nonetheless, an analogous result still holds.

B Gaussian projections

In this section we describe and analyze a simple decoding algorithm for Gaussian projections.

Theorem B.1. *Let $\mathbf{x} \in \mathbb{R}^d$, and let G be a random Gaussian matrix in $\mathbb{R}^{d' \times d}$. Then for any i, there exists a linear function f_i such that*

$$\mathbf{E}_G\left[(f_i(G\mathbf{x}) - x_i)^2\right] \leq \frac{\|\mathbf{x} - x_i \mathbf{e}_i\|_2^2}{d'}$$

Proof. Recall that $G \in \mathbb{R}^{d' \times d}$ is a random Gaussian matrix where each entry is chosen i.i.d. from $N(0, 1/d')$. For any i, conditioned on $G_{ji} = g_{ji}$, we have that the random variable $Y_j | G_{ji} = g_{ji}$ is distributed according to the following distribution,

$$\begin{aligned}(Y_j | G_{ji} = g_{ji}) &\sim g_{ji} x_i + \sum_{i' \neq i} G_{ji'} x_{i'} \\ &\sim g_{ji} x_i + N(0, \|\mathbf{x} - x_i \mathbf{e}_i\|_2^2 / d') \,.\end{aligned}$$

Consider a linear estimator for x_i:

$$\hat{x}_i \overset{def}{=} \frac{\sum_j \alpha_{ji} (y_j / g_{ji})}{\sum_j \alpha_{ji}} \,,$$

for some non-negative α_{ji}'s. It is easy to verify that for any vector of α's, the expectation of $\hat{x}_i$, when taken over the random choices of $G_{ji'}$ for $i' \neq i$, is x_i. Moreover, the variance of $\hat{x}_i$ is

$$\frac{\|\mathbf{x} - x_i \mathbf{e}_i\|_2^2}{d} \frac{\sum_j (\alpha_{ji} / g_{ji})^2}{(\sum_j \alpha_{ji})^2} \,.$$

Minimizing the variance of $\hat{x}_i$ w.r.t α_{ji}'s gives us $\alpha_{ji} \propto g_{ji}^2$. Indeed, the partial derivatives are,

$$\frac{\partial \left(\sum_j (\alpha_{ji}/g_{ji})^2 - \lambda \sum_j \alpha_{ji} \right)}{\partial \alpha_{ji}} = \frac{2\alpha_{ji}}{g_{ji}^2} - \lambda \,,$$

which is zero at $\alpha_{ji} = \lambda g_{ji}^2 / 2$. This choice of α_{ji}'s translates to

$$\mathbf{E}[(\hat{x}_i - x_i)^2] = \frac{\|\mathbf{x} - x_i \mathbf{e}_i\|_2^2}{d'} \frac{1}{\sum_j g_{ji}^2} \,,$$

which in expectation, now taken over the choices of g_{ji}'s, is at most $\|\mathbf{x} - x_i \mathbf{e}_i\|_2^2 / d'$. Thus, the claim follows. ☐

For comparison, if $\mathbf{x} \in B_{d,k}$, then the expected error in estimating x_i is $\sqrt{\frac{k-1}{d'}}$, so that taking $d' = (k-1) \log \frac{1}{\delta} / \varepsilon^2$ suffices to get a error ε estimate of any fixed bit with probablity $1 - \delta$. Setting $\varepsilon = \frac{1}{2}$, we can recover x_i with probability $1 - \delta$ for $\mathbf{x} \in B_{d,k}$ with $d = 4(k-1) \log \frac{1}{\delta}$. This implies Theorem B.2. However, note that the decoding layer is now densely connected to the input layer. Moreover, for a k-sparse vectors $\mathbf{x}$ that are is necessarily binary, the error grows with the 2-norm of the vector $\mathbf{x}$, and can be arbitrarily larger than that for the sparse sketch. Note that $G\,x$ still contains sufficient information to recover $\mathbf{x}$ with error depending only on $\|\mathbf{x} - head_k(\mathbf{x})\|_2$. To our knowledge, all the known decoders are adaptive algorithms, and we leave open the question of whether bounds depending on the 2-norm of the residual $(\mathbf{x} - head_k(\mathbf{x}))$ are achievable by neural networks of small depth and complexity.

Theorem B.2. *For every $\mathbf{w} \in \mathcal{H}_{d,s}$ there exists a set of weights for a network $N \in \mathcal{N}_s(\text{Relu})$ such that for each $\mathbf{x} \in B_{d,k}$,*

$$Pr_{h_{1:t}}[N(G\mathbf{x})) = \mathbf{w}^\top \mathbf{x}] \geq 1 - \delta \,,$$

as long as G is a random $m \times d$ Gaussian matrix, with $m \geq 4k \log(s/\delta)$.

C Deterministic Sketching

First we show that if we allow large scalars in the sketches, we can construct a deterministic $(2k+1)$-dimensional sketch from which a shallow network can reconstruct any monomial.

We will also show a lower bound of k on the required dimensionality.

For every $\mathbf{x} \in B_{d,k}$ define a degree $2k$ univariate real polynomial by,

$$p_{\mathbf{x}}(z) \quad = \quad 1 - (k+1) \prod_{\{i | x_i = 1\}} (z - i)^2.$$

It is easy to verify that this construction satifies the following.

Claim C.1. *Suppose that* $\mathbf{x} \in B_{d,k}$*, and let* $p_{\mathbf{x}}(\cdot)$ *be defined as above. If* $x_j = 1$*, then* $p_{\mathbf{x}}(j) = 1$*. If* $x_j = 0$*, then* $p_{\mathbf{x}}(j) \leq -k$*.*

Let the coeffients of $p_{\mathbf{x}}(z)$ be $a_i(\mathbf{x})$ so that

$$p_{\mathbf{x}}(z) \overset{def}{=} \sum_{i=0}^{2k} a_i(\mathbf{x}) z^i \,.$$

Define the deterministic sketch $DSk_{d,k} : B_{d,k} \to \mathbb{R}^{2k+1}$ as follows,

$$DSk_{d,k}(\mathbf{x}) = (a_{\mathbf{x},0}, \ldots, a_{\mathbf{x},2k}) \tag{3}$$

For a non-empty subset $A \subset [d]$ and $\mathbf{y} \in \mathbb{R}^{2k+1}$ define

$$DecPoly_{d,k}(\mathbf{y}, A) = \frac{\sum_{j \in A} \sum_{i=0}^{2k} y_i j^i}{|A|} \,. \tag{4}$$

Theorem C.2. *For every* $\mathbf{x} \in B_{d,k}$ *and a non-empty set* $A \subset [d]$ *we have*

$$\prod_{j \in A} x_j = \mathrm{Relu}(DecPoly_{d,k}(DSk_{d,k}(\mathbf{x}), A)) \,.$$

Proof. We have that

$$\begin{aligned} DecPoly_{d,k}(DSk_{d,k}(\mathbf{x}), A) &= \frac{\sum_{j \in A} \sum_{i=0}^{2k} a_{\mathbf{x},i} j^i}{|A|} \\ &= \frac{\sum_{j \in A} p_{\mathbf{x}}(j)}{|A|} \,. \end{aligned}$$

In words, the decoding is the average value of $p_{\mathbf{x}}(j)$ over the indices $j \in A$. Now first suppose that $\prod_{j \in A} x_j = 1$. Then for each $j \in A$ we have $x_j = 1$ so that by Claim C.1, $p_{\mathbf{x}}(j) = 1$. Thus the average $DecPoly_{d,k}(DSk_{d,k}(\mathbf{x}), A) = 1$.

On the other hand, if $\prod_{j \in A} x_j = 0$ then for some $j \in A$, say j^*, $x_{j^*} = 0$. In this case, Claim C.1 implies that $p_{\mathbf{x}}(j^*) \leq -k$. For every other j, $p_{\mathbf{x}}(j) \leq 1$, and each $p_{\mathbf{x}}(j)$ is non-negative only when $x_j = 1$, which happens for at most k indices j. Thus the sum over non-negative $p_{\mathbf{x}}(j)$ can be no larger than k. Adding $p_{\mathbf{x}}(j^*)$ gives us zero, and any additional j's can only further reduce the sum. Thus the average in non-positive and hence the Relu is zero, as claimed. □

The last theorem shows that $B_{d,k}$ can be sketched in $\mathbb{R}^q$, where $q = 2k + 1$, such that arbitrary products of variables be decoded by applying a linear function followed by a ReLU. It is natural to ask what is the smallest dimension q for which such a sketch exists. The following theorem shows that q must be at least k. In fact, this is true even if we only require to decode single variables.

Theorem C.3. *Let* $Sk : B_{d,k} \to \mathbb{R}^q$ *be a mapping such that for every* $i \in [d]$ *there is* $\mathbf{w}_i \in \mathbb{R}^q$ *satisfying* $x_i = \mathrm{Relu}(\langle \mathbf{w}_i, Sk(\mathbf{x}) \rangle)$ *for each* $\mathbf{x} \in B_{d,k}$*, then* q *is at least* k*.*

Proof. Denote $X = \{\mathbf{w}_1, \ldots, \mathbf{w}_d\}$ and let $\mathcal{H} \subset \{0,1\}^X$ be the function class consisting of all functions of the form $h_{\mathbf{x}}(\mathbf{w}_i) = \mathrm{sign}(\langle \mathbf{w}_i, Sk(\mathbf{x}) \rangle)$ for $\mathbf{x} \in B_{d,k}$. On one hand, $\mathcal{H}$ is a sub-class of the class of linear separators over $X \subset \mathbb{R}^q$, hence $\mathrm{VC}(\mathcal{H}) \leq q$. On the other hand, we claim that $\mathrm{VC}(\mathcal{H}) \geq k$ which establishes the proof. In order to prove the claim it suffices to show that the set $A = \{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$ is shattered. Let $B \subset A$ let $\mathbf{x}$ be the indicator vector of B. We claim that the restriction of $h_{\mathbf{x}}$ to A is the indicator function of B. Indeed, we have that,

$$\begin{aligned} h_{\mathbf{x}}(\mathbf{w}_i) &= \mathrm{sign}(\langle \mathbf{w}_i, Sk(\mathbf{x}) \rangle) \\ &= \mathrm{sign}(\mathrm{Relu}(\langle \mathbf{w}_i, Sk(\mathbf{x}) \rangle)) \\ &= x_i \\ &= \mathbb{1}[i \in B] \end{aligned}$$

□

We would like to note that both the endcoding and the decoding of determinitic sketched can be computed efficiently, and the dimension of the sketch is smaller than the dimension of a random sketch. We get the following corollaries.

Corollary C.4. *For every* $\mathbf{w} \in \mathcal{H}_{d,s}$ *there exists a set of weights for a network* $N \in \mathcal{N}_s(\mathrm{Relu})$ *such that for each* $\mathbf{x} \in B_{d,k}$, $N(DSk_{d,k}(\mathbf{x})) = \mathbf{w}^\top \mathbf{x}$.

Corollary C.5. *Given* $\mathbf{w} \in \mathbb{R}^s$, *and sets* $A_1, \ldots, A_s \subseteq [d]$, *let* $g : \{0,1\}^d \to \mathbb{R}$ *denote the polynomial function*

$$g(\mathbf{x}) = \sum_{j=1}^{s} w_j \prod_{i \in A_j} x_i = \sum_{j=1}^{s} w_j \bigwedge_{i \in A_j} x_i \,.$$

For any such g, *there exists a set of weights for a network* $N \in \mathcal{N}_s(\mathrm{Relu})$ *such that for each* $\mathbf{x} \in B_{d,k}$, $N(DSk_{h_{1:t}}(\mathbf{x})) = g(\mathbf{x})$.

Known lower bounds for compressed sensing Ba et al. (2010) imply that any linear sketch has size at least $\Omega(k \log \frac{d}{k})$ to allow stable recovery. We leave open the question of whether one can get the compactness and decoding properties of our (non-linear) sketch while ensuring stability.

D Lower Bound for Proper Learning

We now show that if one does not expand the hypothesis class, then even in the simplest of settings of linear classifiers over 1-sparse vectors, the required dimensionality of the projection is much larger than the dimension needed for improper learning. As stated earlier, the result is likely folklore and we present a proof for completeness.

Theorem D.1. *Suppose that there exists a distribution over maps* $\phi : B_{d,1} \to \mathbb{R}^q$ *and* $\psi : B_{d,s} \to \mathbb{R}^q$ *such that for any* $\mathbf{x} \in B_{d,1}, \mathbf{w} \in B_{d,s}$,

$$\Pr\left[sgn\left(\mathbf{w}^\top \mathbf{x} - \frac{1}{2}\right) = sgn\left(\psi(\mathbf{w})^\top \phi(\mathbf{x})\right)\right] \geq \frac{9}{10}\,,$$

where the probability is taken over sampling ϕ *and* ψ *from the distribuion. Then* q *is* $\Omega(s)$.

Proof. If the error is zero, a lower bound on q would follow from standard VC dimension arguments. Concretely, the hypothesis class consisting of $h_{\mathbf{w}}(\mathbf{x}) = \{\mathbf{e}_i^\top \mathbf{x} : w_i = 1\}$ for all $\mathbf{w} \in B_{d,s}$ shatters the set $\{\mathbf{e}_1, \ldots, \mathbf{e}_s\}$. If $sgn(\psi(\mathbf{w})^\top \phi(\mathbf{e}_i)) = sgn(\mathbf{w}^\top \mathbf{x} - \frac{1}{2})$ for each $\mathbf{w}$ and $\mathbf{e}_i$, then the points $\phi(\mathbf{e}_i), i \in [s]$ are shattered by $h_{\psi(\mathbf{w})}(\cdot)$ where $\mathbf{w} \in B_{d,s}$, which is a subclass of linear separators in $\mathbb{R}^q$. Since linear separators in $\mathbb{R}^q$ have VC dimension q, the largest shattered set is no larger, and thus $q \geq s$.

To handle errors, we will use the Sauer-Shelah lemma and show that the set $\{\phi(\mathbf{e}_i) : i \in [s]\}$ has many partitions. To do so, sample ϕ, ψ from the distribution promised above and consider the set of points $A = \{\phi(\mathbf{e}_1), \phi(\mathbf{e}_2), \ldots, \phi(\mathbf{e}_s)\}$. Let $W = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_\kappa\}$ be a set of κ vectors in $B_{d,s}$ such that,

(a) $\mathcal{S}(\mathbf{w}_i) \subseteq [s]$

(b) *Distance property* holds: $|\mathcal{S}(\mathbf{w}_i) \triangle \mathcal{S}(\mathbf{w}_j)| \geq \frac{s}{4}$ for $i \neq j$.

Such a collection of vectors, with $\kappa = 2^{cs}$ for a positive constant c, can be shown to exist by a probabilistic argument or by standard constructions in coding theory. Let $H = \{h_{\psi(\mathbf{w})} : \mathbf{w} \in W\}$ be the linear separators defined by $\psi(\mathbf{w}_i)$ for $i \in W$. For brevity, we denote $h_{\psi(\mathbf{w}_j)}$ by h_j. We will argue that H induces many different subsets of A.

Let $A_j = \{\mathbf{y} \in A : h_j(\mathbf{x}) = 1\} = \{\mathbf{x} \in A : sgn(\psi(\mathbf{w}_j)^\top \mathbf{x}) = 1\}$. Let $E_j \subseteq A$ be the positions where the embeddings ϕ, ψ fail, that is,

$$E_j = \{\phi(\mathbf{e}_i) : i \in [s], sgn(\mathbf{w}_j^\top \mathbf{e}_i - \frac{1}{2}) \neq sgn(\psi(\mathbf{w}_j)^\top \phi(\mathbf{e}_i)\}.$$

Thus $A_j = \mathcal{S}(w_j) \triangle E_j$. By assumption, $\mathbf{E}[|E_j|] \le \frac{s}{10}$ for each j, where the expectation is taken over the choice of ϕ, ψ. Thus $\sum_j \mathbf{E}[|E_j|] \le s\kappa/10$. Renumber the h_j's in increasing order of $|E_j|$ so that $|E_1| \le |E_2| \le \ldots \le |E_\kappa|$. Due to the E_j's being non-empty, not all A_j's are necessarily distinct. Call a $j \in [\kappa]$ *lost* if $A_j = A_{j'}$ for some $j' \le j$.

By definition, $A_j = \mathcal{S}(\mathbf{w}_j) \triangle E_j$. If $A_j = A_{j'}$, then the distance property implies that $E_j \triangle E_{j'} \ge \frac{s}{4}$. Since the E_j's are increasing in size, it follows that for any lost j, $|E_j| \ge \frac{s}{8}$. Thus in expectation at most $4\kappa/5$ of the j's are lost. It follows that there is a choice of ϕ, ψ in the distribution for which H induces $\kappa/5$ distinct subsets of A. Since the VC dimension of H is at most q, the Sauer-Shelah lemma says that

$$\sum_{t \le q} \binom{s}{t} \ge \frac{\kappa}{5} = \frac{2^{cs}}{5}\,.$$

This implies that $q \ge c's$ for some absolute constant c'. $\square$

Note that for the setting of the above example, once we scale the $\mathbf{w}_j$'s to be unit vectors, the margin is $\Theta(\frac{1}{\sqrt{s}})$. Standard results then imply that projecting to $\frac{1}{\gamma^2} = \Theta(s)$ dimension suffices, so the above bound is tight.

For this setting, Theorem 4.1 implies that a sketch of size $O(\log(s/\delta))$ suffices to correctly classify $1-\delta$ fraction of the examples if one allows improper learning as we do.

E Neural Nets on Boolean Inputs

In this short section show that for boolean inputs (irrespective of sparsity), any polynomial with s monomials can be represented by a neural network with one hidden layer of s hidden units. Our result is a simple improvement of Barron's theorem (1993; 1994), for the special case of sparse polynomial functions on 0-1 vectors. In contrast, Barron's theorem, which works for arbitrary inputs, would require a neural network of size $d \cdot s \cdot p^{O(p)}$ to learn an s-sparse degree-p polynomial. The proof of the improvement is elementary and provided for completeness.

Theorem E.1. *Let $\mathbf{x} \in \{0,1\}^d$, and let $g : \{0,1\}^d \to \mathbb{R}$ denote the polynomial function*

$$g(\mathbf{x}) = \sum_{j=1}^{s} w_j \prod_{i \in A_j} x_i = \sum_{j=1}^{s} w_j \bigwedge_{i \in A_j} x_i\,.$$

Then there exists a set of weights for a network $N \in \mathcal{N}_s(\mathrm{Relu})$ such that for each $\mathbf{x} \in \{0,1\}^d$, $N(\mathbf{x}) = g(\mathbf{x})$. Moreover, the weights coming into each node in the hidden layer are in $\{0,1\}$.

Proof. The jth hidden unit implements $h_j = \prod_{i \in A_j} x_i$. As before, for boolean inputs, one can compute h_j as $\mathrm{Relu}(\sum_{i \in A_j} x_i - |A_j| + 1)$. The output node computes $\sum_j w_j h_j$ where h_j is the output of jth hidden unit. $\square$